Chapter 4

Estimation of distribution algorithms

'That is what learning is. You suddenly understand something you've understood all your life, but in a new way. '

Doris Lessing

4.1 Introduction

Generally speaking, all the search strategy types can be classified either as complete or heuristic strategies. The difference between them is that complete strategies perform a systematic examination of all possible solutions of the search space whereas heuristic strategies only concentrate on a part of them following a known algorithm.

Heuristic strategies are also divided in two groups: deterministic and non-deterministic strategies [Pearl, 1984]. The main characteristic of deterministic strategies is that under the same conditions the same solution is always obtained. Examples of this type are forward, backward, stepwise, hill-climbing, threshold accepting, and other well known algorithms, and their main drawback is that they have the risk of getting stuck in local maximum values. Non-deterministic searches are able to escape from these local maxima by means of the randomness [Zhigljavsky, 1991] and, due to their stochasticity, different executions might lead to different solutions under the same conditions.

Some of the stochastic heuristic searches such as simulated annealing only store one solution at every iteration of the algorithm. The stochastic heuristic searches that store more than one solution every iteration (or every generation as each iteration is usually called in these cases) are grouped under the term of population-based heuristics, an example of which is evolutionary computation. In these heuristics, each of the solutions is called *individual*. The group of individuals (also known as *population*) evolves towards more promising areas of the search space while the algorithm carries on with the next generation. Examples of evolutionary computation are Genetic Algorithms (GAs) [Goldberg, 1989, Holland, 1975], evolutionary strategies (ESs) [Rechenberg, 1973], evolutionary programming [Fogel, 1962] and genetic programming [Koza, 1992]. See [Bäck, 1996] for a review on evolutionary algorithms.

The behavior of evolutionary computation algorithms such as GAs depends to a large extent on associated parameters like operators and probabilities of crossing and mutation, size of the population, rate of generational reproduction, the number of generations, and so on. The researcher requires experience in the resolution and use of these algorithms in order to choose the suitable values for these parameters. Furthermore, the task of selecting the best choice of values for all these parameters has been suggested to constitute itself an additional optimization problem [Grefenstette, 1986]. Moreover, GAs show a poor performance in some problems[1] in which the existing operators of crossing and mutation do not guarantee that the building block hypothesis is preserved[2].

All these reasons have motivated the creation of a new type of algorithms classified under the name of Estimation of Distribution Algorithms (EDAs) [Larrañaga and Lozano, 2001, Mühlenbein and Paaß, 1996], trying to make easier to predict the movements of the populations in the search space as well as to avoid the need for so many parameters. These algorithms are also based on populations that evolve as the search progresses and, as well as genetic algorithms, they have a theoretical foundation on probability theory. In brief, EDAs are population-based search algorithms based on probabilistic modelling of promising solutions in combination with the simulation of the induced models to guide their search.

In EDAs the new population of individuals is generated without using neither crossover nor mutation operators. Instead, the new individuals are sampled starting from a probability distribution estimated from the database containing only selected individuals from the previous generation. At the same time, while in other heuristics from evolutionary computation the interrelations between the different variables representing the individuals are kept in mind implicitly (e.g. building block hypothesis), in EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individuals selected at each iteration. In fact, the task of estimating the joint probability distribution associated with the database of the selected individuals from the previous generation constitutes the hardest work to perform. In particular, the latter requires the adaptation of methods to learn models from data that have been developed by researchers in the domain of probabilistic graphical models.

The underlying idea of EDAs will be introduced firstly for the discrete domain, and then it will be reviewed for the continuous domain. As an illustrative example, we will consider the problem that arises in supervised classification known as *feature subset selection* (FSS) [Inza et al., 2000, 2001]. Given a file of cases with information on n predictive variables, $X_1, X_2, \ldots, X_n$, and the class variable C to which the case belongs, the problem consists in selecting a subset of variables that will induce a classifier with the highest predictive capacity in a test set. The cardinality of the search space for this problem is 2^n.

Figure 4.1 shows a generic schematic of EDA approaches, which follow essentially the following steps:

1. Firstly, the initial population D_0 of R individuals is generated. The generation of these R individuals is usually carried out by assuming a uniform distribution on each variable, and next each individual is evaluated.

2. Secondly, in order to make the $l-1^{th}$ population D_{l-1} evolve towards the next D_l one, a number N ($N < R$) of individuals are selected from D_{l-1} following a criterion. We denote by D_{l-1}^N the set of N selected individuals from generation $l-1$.

[1]Problems in which GAs behave worse than simpler search algorithms are known as *deceptive problems*, in which the GAs get usually stuck in local minima and return worse results.

[2]The building block hypothesis [Holland, 1975] states that GAs find solutions by first finding as many building blocks as possible, and then combining them together to give the highest fitness. Following this hypothesis, we can search more effectively by exploiting similarities in the solutions.

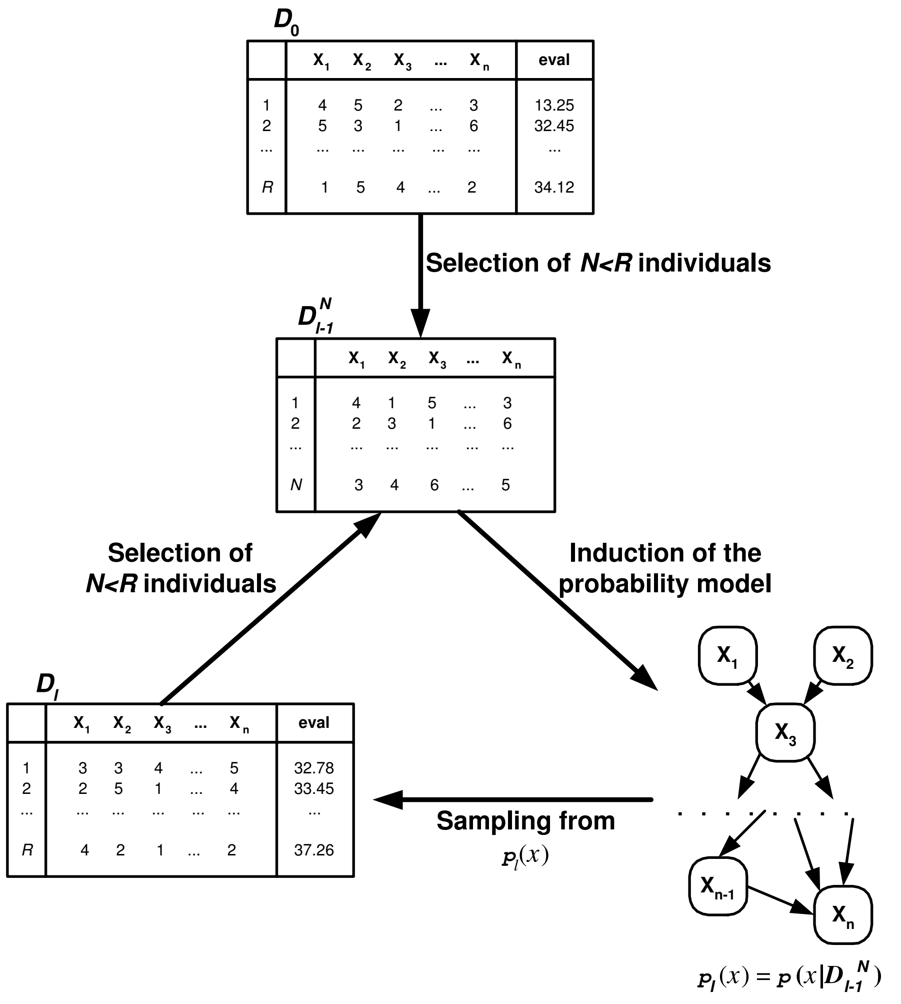


Figure 4.1: Illustration of EDA approaches in the optimization process.

3. Thirdly, the n–dimensional probabilistic model that better represents the interdependencies between the n variables is induced. This step is also known as the *learning* procedure, and it is the most crucial one, since representing appropriately the dependencies between the variables is essential for a proper evolution towards fitter individuals.

4. Finally, the new population D_l constituted by R new individuals is obtained by carrying out the simulation of the probability distribution learned in the previous step. Usually an elitist approach is followed, and therefore the best individual of population D^N_{l-1} is kept in D_l. In this latter case, a total of $R-1$ new individuals is created every generation instead of R.

Steps 2, 3 and 4 are repeated until a stopping condition is verified. Examples of stopping conditions are: achieving a fixed number of populations or a fixed number of different evaluated individuals, uniformity in the generated population, and the fact of not obtaining an individual with a better fitness value after a certain number of generations.

4.2 Probabilistic graphical models

4.2.1 Bayesian networks

This section will introduce the probabilistic graphical model paradigm [Howard and Matheson, 1981, Lauritzen, 1996, Pearl, 1988] that has extensively been used during the last decade as a popular representation for encoding uncertainty knowledge in expert systems [Heckerman and Wellman, 1995]. Only probabilistic graphical models of which a structural part is a directed acyclic graph will be considered, as these adapt properly to EDAs. The following is an adaptation of the paper [Heckerman and Geiger, 1995], and will be used to introduce Bayesian networks as a probabilistic graphical model suitable for its application in EDAs.

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a set of random variables, and let x_i be a value of X_i, the i^{th} component of $\boldsymbol{X}$. Let $\boldsymbol{y} = (x_i)_{X_i \in \boldsymbol{Y}}$ be a value of $\boldsymbol{Y} \subseteq \boldsymbol{X}$. Then, a probabilistic graphical model for $\boldsymbol{X}$ is a graphical factorization of the joint generalized probability density function, $\rho(\boldsymbol{X} = \boldsymbol{x})$ (or simply $\rho(\boldsymbol{x})$). The representation of this model is given by two components: a structure and a set of local generalized probability densities.

The structure S for $\boldsymbol{X}$ is a directed acyclic graph (DAG) that describes a set of conditional independences[3] [Dawid, 1979] about the variables on $\boldsymbol{X}$. $\boldsymbol{Pa}_i^S$ represents the set of parents –variables from which an arrow is coming out in S– of the variable X_i in the probabilistic graphical model which structure is given by S. The structure S for $\boldsymbol{X}$ assumes that X_i and its non descendants are independent given $\boldsymbol{Pa}_i^S$, $i = 2, \ldots, n$. Therefore, the factorization can be written as follows:

$$\rho(\boldsymbol{x}) = \rho(x_1, \ldots, x_n) = \prod_{i=1}^{n} \rho(x_i \mid \boldsymbol{pa}_i^S). \tag{4.1}$$

Furthermore, the local generalized probability densities associated with the probabilistic graphical model are precisely the ones appearing in Equation 4.1.

A representation of the models of the characteristics described above assumes that the local generalized probability densities depend on a finite set of parameters $\boldsymbol{\theta}_S \in \boldsymbol{\Theta}_S$, and as a result the previous equation can be rewritten as follows:

$$\rho(\boldsymbol{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^{n} \rho(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \tag{4.2}$$

where $\boldsymbol{\theta}_S = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$.

After having defined both components of the probabilistic graphical model, and taking them into account, the model itself can be represented by $M = (S, \boldsymbol{\theta}_S)$.

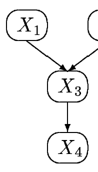
In the particular case of every variable $X_i \in \boldsymbol{X}$ being discrete, the probabilistic graphical model is called *Bayesian network*. If the variable X_i has r_i possible values, $x_i^1, \ldots, x_i^{r_i}$, the local distribution, $p(x_i \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i)$ is an unrestricted discrete distribution:

$$p(x_i{}^k \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i) = \theta_{x_i^k \mid \boldsymbol{pa}_i^j} \equiv \theta_{ijk} \tag{4.3}$$

where $\boldsymbol{pa}_i^{1,S}, \ldots, \boldsymbol{pa}_i^{q_i,S}$ denotes the values of $\boldsymbol{Pa}_i^S$, that is the set of parents of the variable X_i in the structure S; q_i is the number of different possible instantiations of the parent variables

[3]Given $\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W}$ three disjoints sets of variables, $\boldsymbol{Y}$ is said to be conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$ when for any $\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}$ the condition $\rho(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{w}) = \rho(\boldsymbol{y} \mid \boldsymbol{w})$ is satisfied. If this is the case, then we will write $I(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{W})$.

Structure



Local probabilities

$\boldsymbol{\theta_1} = (\theta_{1-1}, \theta_{1-2})$	$p(x_1^1), p(x_1^2)$
$\boldsymbol{\theta_2} = (\theta_{2-1}, \theta_{2-2}, \theta_{2-3})$	$p(x_2^1), p(x_2^2), p(x_2^3)$
$\boldsymbol{\theta_3} = (\theta_{311}, \theta_{321}, \theta_{331},$	$p(x_3^1 \mid x_1^1, x_2^1), p(x_3^1 \mid x_1^1, x_2^2), p(x_3^1 \mid x_1^1, x_2^3),$
$\theta_{341}, \theta_{351}, \theta_{361},$	$p(x_3^1 \mid x_1^2, x_2^1), p(x_3^1 \mid x_1^2, x_2^2), p(x_3^1 \mid x_1^2, x_2^3),$
$\theta_{312}, \theta_{322}, \theta_{332},$	$p(x_3^2 \mid x_1^1, x_2^1), p(x_3^2 \mid x_1^1, x_2^2), p(x_3^2 \mid x_1^1, x_2^3),$
$\theta_{342}, \theta_{352}, \theta_{362})$	$p(x_3^2 \mid x_1^2, x_2^1), p(x_3^2 \mid x_1^2, x_2^2), p(x_3^2 \mid x_1^2, x_2^3)$
$\boldsymbol{\theta_4} = (\theta_{411}, \theta_{421}, \theta_{412}, \theta_{422})$	$p(x_4^1 \mid x_3^1), p(x_4^1 \mid x_3^2), p(x_4^2 \mid x_3^1), p(x_4^2 \mid x_3^2)$

Factorization of the joint mass-probability

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3 \mid x_1, x_2)p(x_4 \mid x_3)$$

Figure 4.2: Structure, local probabilities and resulting factorization in a Bayesian network for four variables (with X_1, X_3 and X_4 having two possible values, and X_2 with three possible values).

of X_i. Thus, $q_i = \prod_{X_g \in \boldsymbol{Pa}_i^S} r_g$. The local parameters are given by $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i})$. In other words, the parameter θ_{ijk} represents the conditional probability that variable X_i takes its k^{th} value, knowing that its parent variables have taken their j^{th} combination of values. We assume that every θ_{ijk} is greater than zero.

Figure 4.2 contains an example of the factorization of a particular Bayesian network with $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$ and $r_2 = 3$, $r_i = 2 \quad i = 1, 3, 4$. From this figure we can conclude that in order to define and build a Bayesian network the user needs to specify:

1.- a structure by means of a directed acyclic graph that reflects the set of conditional independencies among the variables,

2.- the prior probabilities for all root nodes (nodes with no predecessors), that is $p({x_i}^k \mid \emptyset, \boldsymbol{\theta}_i)$ (or θ_{i-k}), and

3.- the conditional probabilities for all other nodes, given all possible combinations of their direct predecessors, $p({x_i}^k \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i)$ (or θ_{ijk}).

4.2.2 Simulation in Bayesian networks

The simulation of Bayesian networks can be regarded as an alternative to exact propagation methods that were developed to reason with networks. This method creates a database with the probabilistic relations between the different variables previous to other procedures. In our particular case, the simulation of Bayesian networks is used merely as a tool to generate new individuals for the next population based on the structure learned previously.

Many approximations to the simulation of Bayesian networks have been developed in recent years. Examples of these are *the likelihood weighting method* developed independently in [Fung and Chang, 1990] and [Shachter and Peot, 1990], and later analyzed in [Shwe and Cooper, 1991], the *backward-forward sampling method* [Fung and del Favero, 1994], the *Markov sampling method* [Pearl, 1987], and the *systematic sampling method* [Bouckaert, 1994]. [Bouckaert et al., 1996] is a good comparison of the previous methods applied to different random Bayesian network models using the average time to execute the algorithm and the average error of the propagation as comparison criteria. Other approaches can be

```
PLS
    Find an ancestral ordering, $\boldsymbol{\pi}$, of the nodes in the Bayesian network
    For $j = 1, 2, \ldots, R$
        For $i = 1, 2, \ldots, n$
            $x_{\pi(i)} \leftarrow$ generate a value from $p(x_{\pi(i)} \mid \boldsymbol{pa}_i)$
```

Figure 4.3: Pseudocode for the Probabilistic Logic Sampling method.

also found in [Chavez and Cooper, 1990, Dagum and Horvitz, 1993, Hryceij, 1990, Jensen et al., 1993].

The method used in this report is the Probabilistic Logic Sampling (PLS) proposed in [Henrion, 1988]. Following this method, the instantiations are done one variable at a time in a forward way, that is, a variable is not sampled until all its parents have already been so. This requires previously to order all the variables from parents to children –any ordering of the variables satisfying such a property is known as ancestral ordering. We will denote $\boldsymbol{\pi} = (\pi(1), \ldots, \pi(n))$ an ancestral order compatible with the structure to be simulated. The concept of forward means that the variables are instantiated from parents to children. For any Bayesian network there is always at least one ancestral ordering since cycles are not allowed in Bayesian networks. Once the values of $\boldsymbol{pa}_i$ –the parent values of a variable X_i– have been assigned, its values are simulated using the distribution $p(x_{\pi(i)} \mid \boldsymbol{pa}_i)$. Figure 4.3 shows the pseudocode of the method.

4.2.3 Gaussian networks

In this section we introduce one example of the probabilistic graphical model paradigm that assumes the joint density function to be a multivariate Gaussian density [Whittaker, 1990].

An individual $\boldsymbol{x} = (x_1, \ldots, x_n)$ in the continuous domain consists of a continuous value in $\Re^n$. The local density function for the i^{th} variable X_i can be computed as the linear-regression model

$$f(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_j - m_j), v_i) \tag{4.4}$$

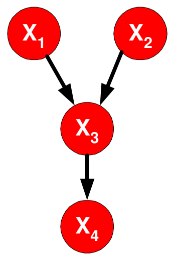
where $\mathcal{N}(x_i; \mu_i, \sigma_i^2)$ is a univariate normal distribution with mean μ_i and variance $v_i = \sigma_i^2$ for the i^{th} variable.

Taking this definition into account, an arc missing from X_j to X_i implies $b_{ji} = 0$ in the former linear-regression model. The local parameters are given by $\boldsymbol{\theta}_i = (m_i, \boldsymbol{b}_i, v_i)$, where $\boldsymbol{b}_i = (b_{1i}, \ldots, b_{i-1i})^t$ is a column vector. A probabilistic graphical model built from these local density functions is known as a *Gaussian network* [Shachter and Kenley, 1989].

The components of the local parameters are as follows: m_i is the unconditional mean of X_i, v_i is the conditional variance of X_i given $\boldsymbol{Pa}_i$, and b_{ji} is a linear coefficient that measures the strength of the relationship between X_j and X_i. Figure 4.4 is an example of a Gaussian network in a 4–dimensional space.

In order to see how Gaussian networks and multivariate normal densities are related, the joint density function of the continuous n–dimensional variable $\boldsymbol{X}$ is by definition a multivariate normal distribution iff:

Structure



Local densities

$$\begin{aligned} \theta_1 &= (m_1, -, \sigma_1^2) & f_{X_1} &\rightsquigarrow \mathcal{N}(x_1; m_1, \sigma_1^2) \\ \theta_2 &= (m_2, -, \sigma_2^2) & f_{X_2} &\rightsquigarrow \mathcal{N}(x_2; m_2, \sigma_2^2) \\ \theta_3 &= (m_3, \boldsymbol{b}_3, \sigma_3^2) & f_{X_3|X_1=x_1,X_2=x_2} &\rightsquigarrow \mathcal{N}(x_3; m_3 + b_{13}(x_1 - m_1) + b_{23}(x_2 - m_2), \sigma_3^2) \\ b_3 &= (b_{13}, b_{23}) && \\ \theta_4 &= (m_4, \boldsymbol{b}_4, \sigma_4^2) & f_{X_4|X_3=x_3} &\rightsquigarrow \mathcal{N}(x_4; m_4 + b_{34}(x_3 - m_3), \sigma_4^2) \\ b_4 &= (b_{34}) && \end{aligned}$$

Factorization of the joint density function

$$f(x_1, x_2, x_3, x_4) = f(x_1) f(x_2) f(x_3|x_1, x_2) f(x_4|x_3) =$$

$$\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{x_1 - m_1}{\sigma_1}\right)^2} \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{x_2 - m_2}{\sigma_2}\right)^2} \frac{1}{\sqrt{2\pi}\sigma_3} e^{-\frac{1}{2}\left(\frac{x_3 - (m_3 + b_{13}(x_1 - m_1) + b_{23}(x_2 - m_2))}{\sigma_3}\right)^2} \frac{1}{\sqrt{2\pi}\sigma_4} e^{-\frac{1}{2}\left(\frac{x_4 - (m_4 + b_{34}(x_3 - m_3))}{\sigma_4}\right)^2}$$

Figure 4.4: Structure, local densities and resulting factorization for a Gaussian network with four variables.

$$f(\boldsymbol{x}) \equiv \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) \equiv (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \tag{4.5}$$

where $\boldsymbol{\mu}$ is the vector of means, Σ is covariance matrix $n \times n$, and $|\Sigma|$ denotes the determinant of Σ. The inverse of this matrix, $W = \Sigma^{-1}$, which elements are denoted by w_{ij}, is known as the precision matrix.

This density can also be written as a product of n conditional densities using the chain rule, namely

$$f(\boldsymbol{x}) = \prod_{i=1}^{n} f(x_i \mid x_1, \ldots, x_{i-1}) = \prod_{i=1}^{n} \mathcal{N}(x_i; \mu_i + \sum_{j=1}^{i-1} b_{ji}(x_j - \mu_j), v_i) \tag{4.6}$$

where μ_i is the unconditional mean of X_i, v_i is the variance of X_i given $X_1, \ldots, X_{i-1}$, and b_{ji} is a linear coefficient reflecting the strength of the relationship between variables X_j and X_i [de Groot, 1970]. This notation allows us to represent a multivariate normal distribution as a Gaussian network, where for any $b_{ji} \neq 0$ with $j < i$ this network will contain an arc from X_j to X_i.

Extending this idea it is also possible to generate a multivariate normal density starting from a Gaussian network. The unconditional means in both paradigms verify that $m_i = \mu_i$ for all $i = 1, \ldots, n$, [Shachter and Kenley, 1989] describe the general transformation procedure to build the precision matrix W of the normal distribution that the Gaussian network represents from its $\boldsymbol{v}$ and $\{b_{ji} \mid j < i\}$. This transformation can be done with the following recursive formula for $i > 0$, and $W(1) = \frac{1}{v_1}$:

$$W(i+1) = \begin{pmatrix} W(i) + \frac{\boldsymbol{b}_{i+1} \boldsymbol{b}_{i+1}^t}{v_{i+1}}, & -\frac{\boldsymbol{b}_{i+1}}{v_{i+1}} \\ -\frac{\boldsymbol{b}_{i+1}^t}{v_{i+1}}, & \frac{1}{v_{i+1}} \end{pmatrix} \tag{4.7}$$

where $W(i)$ denotes the $i \times i$ upper left submatrix, $\boldsymbol{b}_i$ is the column vector $(b_{1i}, \ldots, b_{i-1i})^t$ and $\boldsymbol{b}_i^t$ is its transposed vector.

For instance, taking into account the example in Figure 4.4 where $X_1 \equiv \mathcal{N}(x_1; m_1, v_1)$, $X_2 \equiv \mathcal{N}(x_2; m_2, v_2)$, $X_3 \equiv \mathcal{N}(x_3; m_3 + b_{13}(x_1 - m_1) + b_{23}(x_2 - m_2), v_3)$ and $X_4 \equiv \mathcal{N}(x_4; m_4 + b_{34}(x_3 - m_3), v_4)$, the procedure described above results in the following precision matrix W:

$$W = \begin{pmatrix} \frac{1}{v_1} + \frac{b_{13}^2}{v_3}, & \frac{b_{13}b_{23}}{v_3}, & -\frac{b_{13}}{v_3}, & 0 \\ \frac{b_{23}b_{13}}{v_3}, & \frac{1}{v_2} + \frac{b_{23}^2}{v_2}, & -\frac{b_{23}}{v_3}, & 0 \\ -\frac{b_{13}}{v_3}, & -\frac{b_{23}}{v_3}, & \frac{1}{v_3} + \frac{b_{34}^2}{v_4}, & -\frac{b_{34}}{v_4} \\ 0, & 0, & -\frac{b_{34}}{v_4}, & \frac{1}{v_4} \end{pmatrix}. \qquad (4.8)$$

The representation of a multivariate normal distribution by means of a Gaussian network is more appropriated for model elicitation and understanding rather than the standard representation, as in the latter it is important to ensure that the assessed covariance matrix is positive–definite. In addition, the latter requires to check that the database D with N cases, $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, follows a multivariate normal distribution.

4.2.4 Simulation in Gaussian networks

In [Ripley, 1987] two general approaches for sampling from multivariate normal distributions were introduced. The first method is based on a Cholesky decomposition of the covariance matrix, and the second, known as the conditioning method, generates instances of $\boldsymbol{X}$ by sampling X_1, then X_2 conditionally to X_1, and so on. This second method is analogous to PLS, the sampling procedure introduced in Section 4.2.2 for Bayesian networks, but with the particularity of being designed for Gaussian networks.

The simulation of a univariate normal distribution can be carried out by means of a simple method based on the sum of 12 uniform variables. Traditional methods based on the ratio-of-uniforms [Box and Muller, 1958, Brent, 1974, Marsaglia et al., 1976] could also be applied alternatively.

4.3 Estimation of distribution algorithms in discrete domains

4.3.1 Introduction

This section introduces the notations that will be used to describe EDAs in discrete domains. It also constitutes a review of the EDA approaches for combinatorial optimization problems that can be found in the literature.

Let X_i $(i = 1, \ldots, n)$ be a random variable. A possible instantiation of X_i will be denoted x_i. $p(X_i = x_i)$ –or simply $p(x_i)$– will denote the probability that the variable X_i takes the value x_i. Similarly, $\boldsymbol{X} = (X_1, \ldots, X_n)$ will represent an n–dimensional random variable, and $\boldsymbol{x} = (x_1, \ldots, x_n)$ one of its possible realizations. The mass probability of $\boldsymbol{X}$ will be denoted by $p(\boldsymbol{X} = \boldsymbol{x})$ –or simply $p(\boldsymbol{x})$. The conditional probability of the variable X_i given the value x_j of the variable X_j will be written as $p(X_i = x_i \mid X_j = x_j)$ (or simply $p(x_i \mid x_j)$). D will denote a data set, i.e., a set of R instantiations of the variables $(X_1, \ldots, X_n)$.

Figure 4.5 shows the pseudocode of EDAs in combinatorial optimization problems using the notation introduced, where $\boldsymbol{x} = (x_1, \ldots, x_n)$ will represent the individuals of n genes, and D_l will denote the population of R individuals in the l^{th} generation. Similarly, D_l^N will represent the population of the selected N individuals from D_{l-1}. In EDAs the main task is to estimate $p(\boldsymbol{x} \mid D_{l-1}^N)$, that is, the joint conditional probability over one individual $\boldsymbol{x}$ being

```
EDA
    $D_0 \leftarrow$ Generate $R$ individuals (the initial population) randomly
    Repeat for $l = 1, 2, \ldots$ until a stopping criterion is met
        $D_{l-1}^N \leftarrow$ Select $N < R$ individuals from $D_{l-1}$ according to
                a selection method
        $p_l(\boldsymbol{x}) = p(\boldsymbol{x} \mid D_{l-1}^N) \leftarrow$ Estimate the probability distribution
                of an individual being among the selected individuals
        $D_l \leftarrow$ Sample $R$ individuals (the new population) from $p_l(\boldsymbol{x})$
```

Figure 4.5: Pseudocode for EDA approaches in discrete domains.

among the selected individuals. This joint probability must be estimated every generation. We will denote by $p_l(\boldsymbol{x}) = p_l(\boldsymbol{x} \mid D_{l-1}^N)$ the joint conditional probability at the l^{th} generation.

The most important step is to find the interdependencies between the variables that represent one point in the search space. The basic idea consists in inducing probabilistic models from the best individuals of the population. Once the probabilistic model has been estimated the model is sampled to generate new individuals (new solutions), which will be used to generate a new model in the next generation. This procedure is repeated until a stopping criterion is satisfied. Moreover, the most difficult step for EDAs is actually to estimate satisfactorily the probability distribution $p_l(\boldsymbol{x})$, as the computation of all the parameters needed to specify the underlying probability model becomes impractical. That is why several approximations propose to factorize the probability distribution according to a probability model.

The next sections introduce EDA approaches that can be found in the literature. All the algorithms and methods are classified depending on the maximum number of dependencies between variables that they can account for (maximum number of parents that a variable X_i can have in the probabilistic graphical model). The reader can find in [Larrañaga and Lozano, 2001] a more complete review of this topic.

4.3.2 Without interdependencies

All methods belonging to this category assume that the n–dimensional joint probability distribution factorizes like a product of n univariate and independent probability distributions, that is $p_l(\boldsymbol{x}) = \prod_{i=1}^{n} p_l(x_i)$. This assumption appears to be inexact due to the nature of any difficult optimization problem, where interdependencies between the variables will exist to some degree. Nevertheless, this approximation can lead to an acceptable behavior of EDAs for some problems like the ones on which independence between variables can be assumed.

There are several approaches corresponding to this category that can be found in the literature. Examples are Bit-Based Simulated Crossover –BSC– [Syswerda, 1993], the Population-Based Incremental Learning –PBIL– [Baluja, 1994], the compact Genetic Algorithm [Harik et al., 1998], and the Univariate Marginal Distribution Algorithm –UMDA– [Mühlenbein, 1998].

As an example to show the different ways of computing $p_l(x_i)$, in UMDA this task is done by estimating the relative marginal frequencies of the i^{th} variable within the subset of selected individuals D^N_{l-1}. We describe here this algorithm in more detail as an example of approaches on this category.

UMDA –Univariate Marginal Distribution Algorithm

This algorithm assumes all the variables to be independent in order to estimate the mass joint probability. More formally, the UMDA approach can be written as:

$$p_l(\boldsymbol{x}; \boldsymbol{\theta}^l) = \prod_{i=1}^{n} p_l(x_i; \boldsymbol{\theta}^l_i) \tag{4.9}$$

where $\boldsymbol{\theta}^l_i = \left(\theta^l_{ijk}\right)$ is recalculated every generation by its maximum likelihood estimation, i.e. $\widehat{\theta}^l_{ijk} = \frac{N^{l-1}_{ijk}}{N^{l-1}_{ij}}$, N^l_{ijk} is the number of cases in which the variable X_i takes the value x^k_i when its parents are on their j^{th} combination of values for the l^{th} generation, with $N^{l-1}_{ij} = \sum_k N^{l-1}_{ijk}$. The latter estimation can be allowed since the representation of individuals chosen assumes that, all the variables are discrete, and therefore the estimation of the local parameters needed to obtain the joint probability distribution –$\widehat{\theta}^{\,l}_{ijk}$– is done by simply calculating the relative marginal frequencies of the i^{th} variable within the subset of selected individuals D^N_{l-1} in the l^{th} generation.

4.3.3 Pairwise dependencies

In an attempt to express the simplest possible interdependencies among variables, all the methods in this category propose that the joint probability distribution can be estimated well and fast enough by only taking into account dependencies between pairs of variables. Figure 4.6 shows examples of graphical models where these pairwise dependencies between variables are expressed.

Algorithms in this category require therefore an additional step that was not required in the previous one, which is the construction of a structure that best represents the probabilistic model. In other words, the parametric learning of the previous category –where the structure of the arc-less model remains fixed– is extended to a structural one.

An example of this second category is the greedy algorithm called MIMIC (Mutual Information Maximization for Input Clustering) proposed in [de Bonet et al., 1997], which is explained in more detail below. Other approaches in this group are the ones proposed in [Baluja and Davies, 1997] and the one called BMDA (Bivariate Marginal Distribution Algorithm) [Pelikan and Mühlenbein, 1999].

MIMIC –Mutual Information Maximization for Input Clustering

MIMIC is an EDA proposed for the first time in [de Bonet et al., 1997]. The main idea is to describe the true mass joint probability as closely as possible by using only one univariate marginal probability and $n - 1$ pairwise conditional probability functions.

Given a permutation $\pi = (i_1, i_2, \ldots, i_n)$, we define the class of probability functions, $\mathcal{P}_\pi(\boldsymbol{x})$, as

$$\mathcal{P}_\pi(\boldsymbol{x}) = \{p_\pi(\boldsymbol{x}) \mid p_\pi(\boldsymbol{x}) = p(x_{i_1} \mid x_{i_2}) \cdot p(x_{i_2} \mid x_{i_3}) \cdot \ldots \cdot p(x_{i_{n-1}} \mid x_{i_n}) \cdot p(x_{i_n})\} \tag{4.10}$$

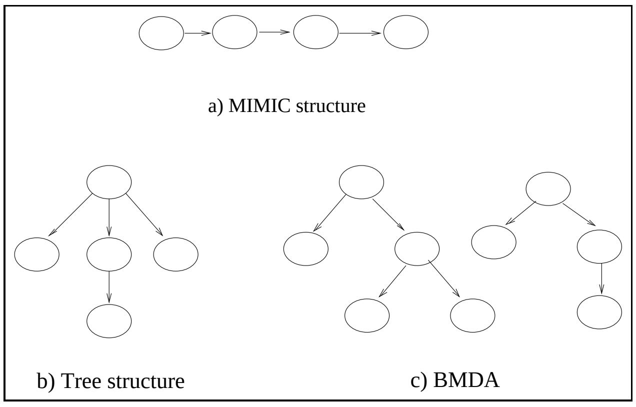


Figure 4.6: Graphical representation of proposed EDA in combinatorial optimization with pairwise dependencies (MIMIC, tree structure, BMDA).

where $p(x_{i_n})$ and $p(x_{i_j} \mid x_{i_{j+1}})$, $j = 1, \ldots, n-1$, are estimated by the marginal and conditional relative frequencies of the corresponding variables within the subset of selected individuals D^N_{l-1} in the l^{th} generation. The goal for MIMIC is to choose the appropriate permutation π^* such that the associated $p_{\pi^*}(\boldsymbol{x})$ minimizes the Kullback-Leibler information divergence between the true probability function, $p(\boldsymbol{x})$, and the probability functions, $p_\pi(\boldsymbol{x})$, of the class $\mathcal{P}_\pi(\boldsymbol{x})$. More formally,

$$D_{K-L}(p(\boldsymbol{x}), p_\pi(\boldsymbol{x})) = \mathsf{E}_{p(\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x})}{p_\pi(\boldsymbol{x})}\right] = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{p_\pi(\boldsymbol{x})}. \tag{4.11}$$

This Kullback-Leibler information divergence can be expressed using the Shanon entropy of a probability function, $h(p(\boldsymbol{x})) = -\mathsf{E}_{p(\boldsymbol{x})}[\log p(\boldsymbol{x})]$, in the following way:

$$\begin{aligned} D_{K-L}(p(\boldsymbol{x}), p_\pi(\boldsymbol{x})) = {} & -h(p(\boldsymbol{x})) + h(X_{i_1} \mid X_{i_2}) + \\ & h(X_{i_2} \mid X_{i_3}) + \ldots + h(X_{i_{n-1}} \mid X_{i_n}) + h(X_{i_n}) \end{aligned} \tag{4.12}$$

where $h(X \mid Y)$ denotes the mean uncertainty in X given Y, that is:

$$h(X \mid Y) = \sum_y h(X \mid Y = y) p_Y(y) \tag{4.13}$$

and

$$h(X \mid Y = y) = -\sum_x p(X = x \mid Y = y) \log p_{X|Y}(x \mid y) \tag{4.14}$$

expresses the uncertainty in X given that $Y = y$.

The latter equation can be rewritten by taking into account that $-h(p(\boldsymbol{x}))$ does not depend on π. Therefore, the task to accomplish is to find the sequence π^* that minimizes the expression

$$J_\pi(\boldsymbol{x}) = h(X_{i_1} \mid X_{i_2}) + \ldots + h(X_{i_{n-1}} \mid X_{i_n}) + h(X_{i_n}). \tag{4.15}$$

In [de Bonet et al., 1997] the authors prove that it is possible to find an approximation of π^* avoiding the need to search over all $n!$ permutations by using a straightforward greedy algorithm. The proposed idea consists in selecting firstly X_{i_n} as the variable with the smallest estimated entropy, and then in successive steps to pick up the variable –from the set of

```
MIMIC - Greedy algorithm to obtain $\pi^*$
    (1) $i_n = \arg\min_j \widehat{h}(X_j)$
        –search for the variable with shortest entropy
    (2) $i_k = \arg\min_j \widehat{h}(X_j \mid X_{i_{k+1}})$
        $j \neq i_{k+1}, \ldots, i_n \quad k = n-1, n-2, \ldots, 2, 1$
        –every step, from all the variables not selected up to that step, look
         for the variable of shortest entropy conditioned to the one before
```

Figure 4.7: MIMIC approach to estimate the mass joint probability distribution.

variables not chosen so far– such that its average conditional entropy with respect to the previous one is the smallest.

Figure 4.7 shows the pseudocode of MIMIC.

4.3.4 Multiple interdependencies

Several other EDA approaches in the literature propose the factorization of the joint probability distribution to be done by statistics of order greater than two. Figure 4.8 shows different probabilistic graphical models that are included in this category. As the number of dependencies between variables is greater than in the previous categories, the complexity of the probabilistic structure as well as the task of finding the best structure that suits the model is bigger. Therefore, these approaches require a more complex learning process.

The following is a brief review of the most important EDA approaches that can be found in the literature within this category:

- The FDA (Factorized Distribution Algorithm) is introduced in [Mühlenbein et al., 1999]. This algorithm applies to additively decomposed functions for which, using the running intersection property, a factorization of the mass-probability based on residuals and separators is obtained.

- In [Etxeberria and Larrañaga, 1999] a factorization of the joint probability distribution encoded by a Bayesian network is learnt from the database containing the selected individuals in every generation. The algorithm developed is called EBNA (Estimation of Bayesian Networks Algorithm), and it makes use of the Bayesian Information Criterion (BIC) score as the measure of the quality of the Bayesian network structure together with greedy algorithms that perform the search in the space of models. This algorithm is explained in more detail later in this section as an example of its category.

- In [Pelikan et al., 1999] the authors propose an algorithm called BOA (Bayesian Optimization Algorithm) which uses a Bayesian metric –the Bayesian Dirichlet equivalent (BDe) [Heckerman et al., 1995]– to measure the goodness of every structure found. A greedy search procedure is also used for this purpose. The search starts in each generation from scratch.

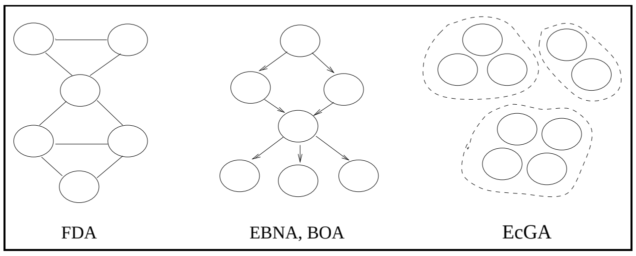


Figure 4.8: Graphical representation of proposed EDA in combinatorial optimization with multiply dependencies (FDA, EBNA, BOA and EcGA).

- The LFDA (Learning Factorized Distribution Algorithm) is introduced in [Mühlenbein and Mahning, 1999], which follows essentially the same approach as in EBNA.
- The Extend compact Genetic Algorithm (EcGA) proposed in [Harik, 1999] is an algorithm of which the basic idea consists in factorizing the joint probability distribution as a product of marginal distributions of variable size.

EBNA –Estimation of Bayesian Network Algorithm

EBNA is an EDA proposed in [Etxeberria and Larrañaga, 1999] that belongs to the category of algorithms that take into account multiple interdependencies between variables. This algorithm proposes the construction of a probabilistic graphical model with no restriction in the number of parents that variables can have.

In brief, the EBNA approach is based on a score+search method: a measure is selected to indicate the adequacy of any Bayesian network for representing the interdependencies between the variables –the score– and this is applied in a procedure that will search for the structure that obtains a satisfactory score value –the search process.

Scores for Bayesian networks.
In this algorithm, given a database D with N cases, $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, a measure of the success of any structure S to describe the observed data D is proposed. This measure is obtained by computing the maximum likelihood estimate –$\widehat{\boldsymbol{\theta}}$– for the parameters $\boldsymbol{\theta}$ and the associated maximized log likelihood, $\log p(D \mid S, \widehat{\boldsymbol{\theta}})$. The main idea in EBNA is to search for the structure that maximizes $\log p(D \mid S, \widehat{\boldsymbol{\theta}})$ using an appropriate search strategy. This is done by scoring each structure by means of its associated maximized log likelihood.

Using the notation introduced in Section 4.3.1, we obtain

$$\begin{aligned}
\log p(D \mid S, \boldsymbol{\theta}) &= \log \prod_{w=1}^{N} p(\boldsymbol{x}_w \mid S, \boldsymbol{\theta}) \\
&= \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i} \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}} \qquad (4.16)
\end{aligned}$$

```
EBNA$_{BIC}$
    $M_0 \leftarrow (S_0, \boldsymbol{\theta}_0)$
    $D_0 \leftarrow$ Sample $R$ individuals from $M_0$
    For $l = 1, 2, \ldots$ until a stop criterion is met
        $D_{l-1}^N \leftarrow$ Select $N$ individuals from $D_{l-1}$
        $S_l^* \leftarrow$ Find the structure which maximizes $BIC(S_l, D_{l-1}^N)$
        $\boldsymbol{\theta}^l \leftarrow$ Calculate $\{\theta_{ijk}^l = \frac{N_{ijk}^{l-1}+1}{N_{ij}^{l-1}+r_i}\}$ using $D_{l-1}^N$ as data set
        $M_l \leftarrow (S_l^*, \boldsymbol{\theta}^l)$
        $D_l \leftarrow$ Sample $R$ individuals from $M_l$ using PLS
```

Figure 4.9: Pseudocode for EBNA$_{BIC}$ algorithm.

where N_{ijk} denotes the number of cases in D in which the variable X_i has the value x_i^k and $\boldsymbol{Pa}_i$ is instantiated as its j^{th} value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

Knowing that the maximum likelihood estimate for θ_{ijk} is given by $\widehat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$, the previous equation can be rewritten as

$$\log p(D \mid S, \widehat{\boldsymbol{\theta}}) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{4.17}$$

For the case of complex models, the sampling error associated with the maximum likelihood estimator might turn out to be too big to consider the maximum likelihood estimate as a reliable value for the parameter –even for a large sample. A common response to this difficulty is to incorporate some form of penalty depending on the complexity of the model into the maximized likelihood. Several penalty functions have been proposed in the literature. A general formula for a penalized maximum likelihood score could be

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - f(N) dim(S) \tag{4.18}$$

where $dim(S)$ is the dimension –i.e. the number of parameters needed to specify the model– of the Bayesian network following the structure given by S. This dimension is computed as $dim(S) = \prod_{i=1}^{n} q_i(r_i - 1)$. The penalization function $f(N)$ is a non-negative one. Examples of values given to $f(N)$ in the literature are the Akaike's Information Criterion (AIC) [Akaike, 1974] –where it is considered as a constant, $f(N) = 1$– and the Jeffreys-Schwarz criterion which is also known as the Bayesian Information Criterion (BIC) [Schwarz, 1978] –where $f(N) = \frac{1}{2} \log N$.

Following the latter criterion, the corresponding BIC score –$BIC(S, D)$– for a Bayesian network structure S constructed from a database D and containing N cases is as follows:

$$BIC(S, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{\log N}{2} \sum_{i=1}^{n} (r_i - 1) q_i \tag{4.19}$$

where N_{ijk} and N_{ij} and q_i are defined as above.

```
EBNA$_{K2}$
    $M_0 \leftarrow (S_0, \boldsymbol{\theta}_0)$
    $D_0 \leftarrow$ Sample $R$ individuals from $M_0$
    For $l = 1, 2, \ldots$ until a stop criterion is met
        $D_{l-1}^N \leftarrow$ Select $N$ individuals from $D_{l-1}$
        $S_l^* \leftarrow$ Find the structure which maximizes $K2(S_l, D_{l-1}^N)$
        $\boldsymbol{\theta}^l \leftarrow$ Calculate $\{\theta_{ijk}^l = \frac{N_{ijk}^{l-1}+1}{N_{ij}^{l-1}+r_i}\}$ using $D_{l-1}^N$ as data set
        $M_l \leftarrow (S_l^*, \boldsymbol{\theta}^l)$
        $D_l \leftarrow$ Sample $R$ individuals from $M_l$ using PLS
```

Figure 4.10: Pseudocode for EBNA$_{K2}$ algorithm.

On the other hand, by assuming that all the local probability distributions θ_{ijk} in EBNA follow a Dirichlet distribution with the hyperparameters $\alpha_{ijk} = 1$, these are calculated every generation using their expected values as obtained in [Cooper and Herskovits, 1992]:

$$\mathsf{E}[\theta_{ijk}^l \mid S, D_{l-1}^N] = \frac{N_{ijk}^{l-1} + 1}{N_{ij}^{l-1} + r_i}. \tag{4.20}$$

The whole approach is illustrated in Figure 4.9, which corresponds to the one developed originally in [Etxeberria and Larrañaga, 1999]. In this paper, the authors use the penalized maximum likelihood as the score to evaluate the goodness of each structure found during the search. In particular, they propose the use of the BIC score. Due to the application of scores other than BIC, the original EBNA that is illustrated in Figure 4.9 is commonly known as EBNA$_{BIC}$.

Another score that has also been proposed in the literature is an adaption of the K2 algorithm [Cooper and Herskovits, 1992], which is also known as EBNA$_{K2}$. Given a Bayesian network, if the cases occur independently, there are no missing values, and the density of the parameters given the structure is uniform, then the authors show that

$$p(D \mid S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{4.21}$$

EBNA$_{K2}$ assumes that an ordering on the variables is available and that, a priori, all structures are equally likely. It searches, for every node, the set of parent nodes that maximizes the following function:

$$g(i, \boldsymbol{Pa}_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{4.22}$$

Following this definition, the corresponding K2 score –$K2(S, D)$– for a Bayesian network structure S constructed from a database D and containing N cases is:

$$K2(S, D) = \sum_{i=1}^{n} g(i, \boldsymbol{Pa}_i) = \sum_{i=1}^{n} \left(\prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right) \tag{4.23}$$

where N_{ijk} and N_{ij} and q_i are defined as above.

EBNA$_{K2}$ is proposed as a *greedy* heuristic. It starts by assuming that a node does not have parents, then in each step it adds incrementally that parent whose addition most increases the probability of the resulting structure. EBNA$_{K2}$ stops adding parents to the nodes when the addition of a single parent can not increase this probability. Obviously, as well as with EBNA$_{BIC}$, this approach does not guarantee to obtain the structure with the highest probability.

Search methods.

Regarding the search method that is combined with the score, in order to obtain the best existing model all possible structures must be searched through. Unfortunately, this has been proved to be NP-hard [Chickering et al., 1994]. Even if promising results have been obtained through global search techniques [Etxeberria et al., 1997a,b, Larrañaga et al., 1996a,b,c], their computation cost makes them impractical for our problem. As the aim is to find a satisfactory model as good as possible –even if not the optimal– in a reasonable period of time, a simpler search method that avoids analyzing all the possible structures is preferred. An example of the latter is the so called B Algorithm [Buntine, 1991]. The B Algorithm is a greedy search heuristic which starts from an arc-less structure and adds iteratively the arcs that produce maximum improvement according to the BIC approximation –although other measures could also be applied. The algorithm stops when adding another arc would not increase the score of the structure.

Local search strategies are another way of obtaining good models. These start from a given structure, and every step the addition or deletion of an arc that improves most the scoring measure is performed. Local search strategies stop when no modification of the structure improves the scoring measure. The main drawback of local search strategies is their heavy dependence on the initial structure. Nevertheless, as [Chickering et al., 1995] showed that local search strategies perform quite well when the initial structure is reasonably good, the model of the previous generation could be used as the initial structure when the search is based on the assumption that $p(\boldsymbol{x} \mid D_l^N)$ will not differ very much from $p(\boldsymbol{x} \mid D_{l-1}^N)$

The initial model M_0 in EBNA is formed by a structure S_0, which is an arc-less DAG, and the local probability distributions given by the n unidimensional marginal probabilities $p(X_i = x_i) = \frac{1}{r_i}$, $i = 1, \ldots, n$ –that is, M_0 assigns the same probability to all individuals. The model of the first generation –M_1– is learnt using Algorithm B, while the rest of the models are learnt by means of a local search strategy which take the model of the previous generation as the initial structure.

4.4 Estimation of distribution algorithms in continuous domains

4.4.1 Introduction

This section complements the previous two ones, as it introduces the EDA algorithms for their use in optimization in continuous domains. The continuous EDAs will be discussed following the same layout as in the previous sections. All the continuous EDA approaches will be also classified using an analogous comparison based on the complexity of the estimation of the probability distribution. In this case, as we are in the continuous domain, the density function will be factorized as a product of n conditional density functions.

The notation for continuous EDAs does not vary significantly from the one presented for discrete EDAs, as continuous EDAs follow essentially equivalent steps to approximate the best solution, and therefore Figure 4.5 is still valid to describe the continuous EDA approach (it would be enough with substituting $p_l(\boldsymbol{x})$ by $f_l(\boldsymbol{x})$). Nevertheless, regarding the learning and simulation steps, EDAs in the continuous domain have some characteristics that make them very particular.

Let $X_i \in \boldsymbol{X}$ be a continuous variable. Similarly as in the discrete domain, a possible instantiation of X_i will be denoted x_i, and D will denote a data set, i.e., a set of R instantiations of the variables $(X_1, \ldots, X_n)$. In the continuous domain, again $\boldsymbol{x} = (x_1, \ldots, x_n)$ represents an individual of n variables, D_l denotes the population of R individuals in the l^{th} generation, and D_l^N represents the population of the selected N individuals from D_l, and as for the discrete case, the main task is still to estimate the joint density function at every generation. We will denote by $f_l(\boldsymbol{x} \mid D_{l-1}^N)$ the joint conditional density function at the l^{th} generation.

The most difficult step in here is also the search of interdependencies between the different variables. Again, in continuous EDAs probabilistic models are induced from the best N individuals of the population. Once the structure is estimated, this model is sampled to generate the R new individuals that will form the new generation. As the estimation of the joint density function is a tedious task, approximations are applied in order to estimate the best joint density function according to the probabilistic model learned at each generation.

As with the discrete domain, all the continuous EDA approaches can be divided in different categories depending on the degree of dependency that they take into account. Following the classification in [Larrañaga and Lozano, 2001], we will divide all the continuous EDA in three main categories.

4.4.2 Without dependencies

This is the category of algorithms that do not take into account dependencies between any of the variables. In this case, the joint density function is factorized as a product of n one-dimensional and independent densities. Examples of continuous EDAs in this category are the Univariate Marginal Distribution Algorithm for application in continuous domains (UMDA$_c$) [Larrañaga et al., 2000], Stochastic Hill-Climbing with Learning by Vectors of Normal Distributions (SHCLVND) [Rudlof and Köppen, 1996], Population-Based Incremental Learning for continuous domains (PBIL$_c$) [Sebag and Ducoulombier, 1998], and the algorithm introduced in [Servet et al., 1997]. As an example of all these, UMDA$_c$ is shown more in detail.

UMDA$_c$

The Univariate Marginal Distribution Algorithm for application in continuous domains (UMDA$_c$) was introduced in [Larrañaga et al., 2000]. In this approach, every generation and for every variable some statistical tests are performed to obtain the density function that best fits the variable. In UMDA$_c$ the factorization of the joint density function is given by

$$f_l(\boldsymbol{x}; \boldsymbol{\theta}^l) = \prod_{i=1}^{n} f_l(x_i, \boldsymbol{\theta}_i^l). \tag{4.24}$$

Unlike UMDA in the discrete case, UMDA$_c$ is a structure identification algorithm meaning that the density components of the model are identified with the aid of hypothesis tests.

```
UMDA$_c$
** learning the joint density function **
for $l = 1, 2, \ldots$ until the stopping criterion is met
  for $i = 1$ to $n$ do
    (i) select via hypothesis test the density function $f_l(x_i; \boldsymbol{\theta}_i^l)$ that
        best fits $D_{l-1}^{N,X_i}$, the projection of the selected individuals over the i$^{th}$ variable
    (ii) obtain the maximum likelihood estimates for $\boldsymbol{\theta}_i^l = (\theta_i^{l,k_1}, \ldots, \theta_i^{l,k_i})$

Each generation the learnt joint density function is expressed as:
  $f_l(\boldsymbol{x}; \boldsymbol{\theta}^l) = \prod_{i=1}^n f_l(x_i, \widehat{\boldsymbol{\theta}}_i^{\,l})$
```

Figure 4.11: Pseudocode to estimate the joint density function followed in UMDA$_c$.

Once the densities have been identified, the estimation of parameters is carried out by means of their maximum likelihood estimates.

If all the univariate distributions are normal distributions, then for each variable two parameters are estimated at each generation: the mean, μ_i^l, and the standard deviation, σ_i^l. It is well known that their respective maximum likelihood estimates are:

$$\widehat{\mu_i}^l = \overline{X_i}^l = \frac{1}{N} \sum_{r=1}^{N} x_{i,r}^l; \qquad \widehat{\sigma_i}^l = \sqrt{\frac{1}{N} \sum_{r=1}^{N} \left(x_{i,r}^l - \overline{X_i}^l \right)^2} \tag{4.25}$$

This particular case of the UMDA$_c$ is called UMDA$_c^G$ (Univariate Marginal Distribution Algorithm for Gaussian models).

Figure 4.11 shows the pseudocode to learn the joint density function followed by UMDA$_c$.

4.4.3 Bivariate dependencies

MIMIC$_c^G$

This algorithm was introduced in [Larrañaga et al., 2000] and is basically an adaptation of the MIMIC algorithm [de Bonet et al., 1997] to the continuous domain. In this, the underlying probability model for every pair of variables is assumed to be a bivariate Gaussian.

Similarly as in MIMIC, the idea is to describe the underlying joint density function that fits the model as closely as possible to the empirical data by using only one univariate marginal density and $n-1$ pairwise conditional density functions. For that, the following theorem [Whittaker, 1990] is used:

Theorem 4.1: [Whittaker, 1990, pp. 167] Let $\boldsymbol{X}$ be a n–dimensional normal density function, $\boldsymbol{X} \equiv \boldsymbol{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \sum)$, then the entropy of $\boldsymbol{X}$ is

$$h(\boldsymbol{X}) = \frac{1}{2} n (1 + \log 2\pi) + \frac{1}{2} \log \mid \sum \mid . \tag{4.26}$$

```
$\text{MIMIC}_c^G$
    Choose $i_n = \arg\min_j \ \hat{\sigma}^2_{X_j}$
    for $k = n-1, n-2, \ldots, 1$
        Choose $i_k = \arg\min_j \ \hat{\sigma}^2_{X_j} - \frac{\hat{\sigma}^2_{X_j X_{i_{k+1}}}}{\hat{\sigma}^2_{X_{i_{k+1}}}}$
        $j \neq i_{k+1}, \ldots, i_n$
```

Figure 4.12: Adaptation of the MIMIC approach to a multivariate Gaussian density function.

When applying this result to univariate and bivariate normal density functions to define MIMIC_c^G, we obtain that

$$h(X) = \frac{1}{2}(1 + \log 2\pi) + \log \sigma_X \tag{4.27}$$

$$h(X \mid Y) = \frac{1}{2}\left[(1 + \log 2\pi) + \log\left(\frac{\sigma_X^2 \sigma_Y^2 - \sigma_{XY}^2}{\sigma_Y^2}\right)\right] \tag{4.28}$$

where $\sigma_X^2(\sigma_Y^2)$ is the variance of the univariate $X(Y)$ variable and σ_{XY} denotes the covariance between the variables X and Y.

The learning of the structure in MIMIC_c^G is shown in Figure 4.12. It follows a straightforward greedy algorithm composed of two steps. In the first one, the variable with the smallest sample variance is chosen. In the second step, the variable X with the smallest estimation of $\frac{\sigma_X^2 \sigma_Y^2 - \sigma_{XY}^2}{\sigma_Y^2}$ regarding the variable Y chosen in the previous iteration is selected, and X is linked to Y in the structure.

4.4.4 Multiple interdependencies

Algorithms in this section are approaches of EDAs for continuous domains in which there is no restriction on the number of interdependencies between variables to take into account on the density function learnt at every generation. In the first example introduced, the density function corresponds to a non restricted multivariate normal density that is learned from scratch at each generation. The next two examples are respectively an adaptation and an improvement of this first model. Finally, this section also introduces edge exclusion test approaches to learn from Gaussian networks, as well as two score+search approaches to search for the most appropriated Gaussian network at each generation.

EMNA$_{global}$

This approach performs the estimation of a multivariate normal density function at each generation. Figure 4.13 shows the pseudocode of EMNA$_{global}$ (Estimation of Multivariate Normal Algorithm – global). In EMNA$_{global}$, at every generation we proceed as follows: the vector of means $\boldsymbol{\mu}_l = (\mu_{1,l}, \ldots, \mu_{n,l})$, and the variance–covariance matrix Σ_l are computed. The elements of the latter are represented as $\sigma^2_{ij,l}$ with $i, j = 1, \ldots, n$. As a result, at every generation all the $2n + \binom{n-1}{2}$ parameters need to be estimated: n means, n variances

```
EMNA$_{global}$
  $D_0 \leftarrow$ Generate $R$ individuals (the initial population) at random
  for $l = 1, 2, \ldots$ until the stopping criterion is met
    $D^N_{l-1} \leftarrow$ Select $N < R$ individuals from $D_{l-1}$ according to the
      selection method
    $f_l(\boldsymbol{x}) = f(\boldsymbol{x} \mid D^N_{l-1}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_l, \Sigma_l) \leftarrow$ Estimate the multivariate
      normal density function from the selected individuals
    $D_l \leftarrow$ Sample $R$ individuals (the new population) from $f_l(\boldsymbol{x})$
```

Figure 4.13: Pseudocode for the EMNA$_{global}$ approach.

and $\begin{pmatrix} n-1 \\ 2 \end{pmatrix}$ covariances. This is performed using their maximum likelihood estimates in the following way:

$$\hat{\mu}_{i,l} = \frac{1}{N}\sum_{r=1}^{N} x^l_{i,r} \quad i = 1, \ldots, n$$

$$\hat{\sigma}^2_{i,l} = \frac{1}{N}\sum_{r=1}^{N}(x^l_{i,r} - \overline{X}^l_i)^2 \quad i = 1, \ldots, n$$

$$\hat{\sigma}^2_{ij,l} = \frac{1}{N}\sum_{r=1}^{N}(x^l_{i,r} - \overline{X}^l_i)(x^l_{j,r} - \overline{X}^l_j) \quad i, j = 1, \ldots, n \quad i \neq j. \tag{4.29}$$

At a first glance the reader could think that this approach requires much more computation than in the other cases in which the estimation of the joint density function is done with Gaussian networks. However, the mathematics on which this approach is based are quite simple. On the other hand, approaches based on edge exclusion tests on Gaussian networks also require the computation of as many parameters as the ones needed by this approach in order to carry out a hypothesis test on them. Moreover, the second type of Gaussian network approaches based on score+search methods also require a lot of extra computation, as the searching process needs to look for the best structure over the whole space of possible models. The reader can find more details about EMNA$_{global}$ in [Larrañaga et al., 2001].

EMNA$_a$

EMNA$_a$ (Estimation of Multivariate Normal Algorithm – adaptive) is an adaptive version of the previous approach.

The biggest particularity of this algorithm is the way of obtaining the first model, $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \Sigma_1)$, the parameters of which are estimated from the best individuals selected in the initial population. After this step, EMNA$_a$ behaves as a steady–step genetic algorithm. The pseudocode for EMNA$_a$ is given in Figure 4.14.

Every iteration, an individual from the current multivariate normal density model is sampled. Next, the goodness of this simulated individual is compared to the worst individual

```
EMNA$_a$
  $D_0 \leftarrow$ Generate $R$ individuals (the initial population) at random
  Select $N < R$ individuals from $D_0$ according to the selection method
  Obtain the first multivariate normal density $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \Sigma_1)$
  for $l = 1, 2, \ldots$ until the stopping criterion is met
      Generate an individual $\boldsymbol{x}_{ge}^l$ from $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_l, \Sigma_l)$
      if $\boldsymbol{x}_{ge}^l$ is better than the worst individual, $\boldsymbol{x}^{l,N}$, then
          1.- Add $\boldsymbol{x}_{ge}^l$ to the population and drop $\boldsymbol{x}^{l,N}$ from it
          2.- Obtain $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{l+1}, \Sigma_{l+1})$
```

Figure 4.14: Pseudocode for the EMNA$_a$ approach.

of the current population. If the fitness value of the new individual has a better value, then the new individual replaces the worst one in the population. In the latter case, it is also necessary to update the parameters of the multivariate normal density function.

The updating of the density function is done using the following formulas that can be obtained by means of simple algebraic manipulations [Larrañaga et al., 2001]:

$$\boldsymbol{\mu}_{l+1} = \boldsymbol{\mu}_l + \frac{1}{N}\left(\boldsymbol{x}_{ge}^l - \boldsymbol{x}^{l,N}\right) \tag{4.30}$$

$$\begin{aligned} \sigma_{ij,l+1}^2 &= \sigma_{ij,l}^2 - \frac{1}{N^2}\left(x_{ge,i}^l - x_i^{l,N}\right) \cdot \sum_{r=1}^{N}\left(x_j^{l,r} - \mu_j^l\right) - \frac{1}{N^2}\left(x_{ge,j}^l - x_j^{l,N}\right) \cdot \sum_{r=1}^{N}\left(x_i^{l,r} - \mu_i^l\right) \\ &+ \frac{1}{N^2}\left(x_{ge,i}^l - x_i^{l,N}\right)\left(x_{ge,j}^l - x_j^{l,N}\right) - \frac{1}{N}\left(x_i^{l,N} - \mu_i^{l+1}\right)\left(x_j^{l,N} - \mu_j^{l+1}\right) \\ &+ \frac{1}{N}\left(x_{ge,i}^l - \mu_i^{l+1}\right)\left(x_{ge,j}^l - \mu_j^{l+1}\right) \end{aligned} \tag{4.31}$$

where $\boldsymbol{x}_{ge}^l$ represents the individual generated in the 1^{th} iteration. Note also that in the EMNA$_a$ approach the size of the population remains constant every generation independently of the fitness of the individual sampled.

EMNA$_i$

EMNA$_i$ (Estimation of Multivariate Normal Algorithm – incremental) is a new approach following a similar idea of EMNA$_a$, as both algorithms generate each generation a simple individual which fitness value is compared to the worst individual in the current population. However, the biggest difference between them is what happens with the worst individual when its fitness value is lower than the new individual: in EMNA$_i$ the worst individual remains in the population, and therefore the population increases in size in those cases. Its main interest is that the rules to update the density function are simpler than in EMNA$_a$.

The reader is referred to [Larrañaga et al., 2001] for more details about this algorithm.

```
EGNA$_{ee}$, EGNA$_{BGe}$, EGNA$_{BIC}$
    For $l = 1, 2, \ldots$ until the stopping criterion is met
        $D_{l-1}^{N} \leftarrow$ Select $N$ individuals from $D_{l-1}$
        (i) $\hat{S}_l \leftarrow$ Structural learning via:
                edge exclusion tests $\rightarrow$ EGNA$_{ee}$
                Bayesian score+search $\rightarrow$ EGNA$_{BGe}$
                penalized maximum likelihood + search $\rightarrow$ EGNA$_{BIC}$
        (ii) $\hat{\boldsymbol{\theta}}^l \leftarrow$ Calculate the estimates for the parameters of $\hat{S}_l$
        (iii) $M_l \leftarrow (\hat{S}_l, \hat{\boldsymbol{\theta}}^l)$
        (iv) $D_l \leftarrow$ Sample $R$ individuals from $M_l$ using the continuous version
                of the PLS algorithm
```

Figure 4.15: Pseudocode for the EGNA$_{ee}$, EGNA$_{BGe}$, and EGNA$_{BIC}$ algorithms.

EGNA$_{ee}$, EGNA$_{BGe}$, EGNA$_{BIC}$

The optimization in continuous domains can also be carried out by means of the learning and simulation of Gaussian networks. An example of this is the EGNA approach (Estimation of Gaussian Networks Algorithm), which is illustrated in Figure 4.15. This approach has three versions: EGNA$_{ee}$ [Larrañaga et al., 2000, Larrañaga and Lozano, 2001], EGNA$_{BGe}$ and EGNA$_{BIC}$ [Larrañaga et al., 2001]. The basic steps of these algorithms each iteration are as follows:

1. Obtain the Gaussian network structure by using one of the different methods proposed, namely edge–exclusion tests, Bayesian score+search, or penalized maximum likelihood+search.

2. Computation of estimates for the parameters of the learned Gaussian network structure.

3. Generation of the Gaussian network model.

4. Simulation of the joint density function expressed by the Gaussian network learned in the previous steps. An adaptation of the PLS algorithm to continuous domains is used for this purpose.

The main difference between all these EGNA approaches is the way of inducing the Gaussian network: in EGNA$_{ee}$ the Gaussian network is induced at each generation by means of edge exclusion tests, while the model induction in the EGNA$_{BGe}$ and EGNA$_{BIC}$ is carried out by score+search approaches. EGNA$_{BGe}$ makes use of a Bayesian score that gives the same value for Gaussian networks reflecting identical conditional (in)dependencies, and EGNA$_{BIC}$ uses a penalized maximum likelihood score based on the Bayesian Information Criterion (BIC). In both EGNA$_{BGe}$ and EGNA$_{BIC}$ a local search is used to search for good structures. These model induction methods are reviewed in the next subsection.

Applications of these EGNA approaches can be found in Cotta et al. (2001), Bengoetxea et al. (2001a), Lozano and Mendiburu (2001) and Robles et al. (2001).

Next, three different methods that can be applied to induce Gaussian networks from data are introduced. The first of them is based on edge exclusion tests, while the other two are score+search methods.

Edge exclusion tests

Dempster (1972) introduced a type of graphical Gaussian models on which the structure of the precision matrix is modelled rather than the variance matrix itself. The aim of this is to simplify the joint n–dimensional normal density by checking if a particular element w_{ij} with $i = 1, \ldots n-1$ and $j > i$ of the $n \times n$ precision matrix W can be set to zero. In [Wermuth, 1976] it was shown that fitting these models is equivalent to check the conditional independence between the corresponding elements of the n–dimensional variable $\boldsymbol{X}$. [Speed and Kiiveri, 1986] showed that this procedure is equivalent to check the possibility of deleting the arc connecting the nodes corresponding to X_i and X_j in the conditional independence graph. That is why these tests are commonly known as edge exclusion tests. As the fact of excluding any edge connecting X_i and X_j is analogous to accepting the null hypothesis $H_0 : w_{ij} = 0$ with the alternative hypothesis $H_A : w_{ij}$ unspecified, many graphical model selection procedures have a first step performing the $\binom{n}{2}$ single edge exclusion tests. In this first step, likelihood ratio statistic is evaluated and compared to a χ^2 distribution. However, the use of this distribution is only asymptotically correct. In [Smith and Whittaker, 1998] the authors introduced an alternative to these tests based on the likelihood ratio test that is discussed next.

The likelihood ratio test statistic to exclude the arc between X_i and X_j from a graphical Gaussian model is defined as $T_{lik} = -n \log(1 - r^2_{ij|rest})$, where $r_{ij|rest}$ is the sample partial correlation of X_i and X_j adjusted for the rest of the variables. This can be expressed in terms of the maximum likelihood estimate for every element of the precision matrix as $r_{ij|rest} = -\hat{w}_{ij}(\hat{w}_{ii}\hat{w}_{jj})^{-\frac{1}{2}}$ [Whittaker, 1990].

In [Smith and Whittaker, 1998] the density and distribution functions of the likelihood ratio test statistic is obtained through the null hypothesis. These expressions are of the form:

$$f_{lik}(t) = g_{\mathcal{X}}(t) + \frac{1}{4}(t-1)(2n+1)g_{\mathcal{X}}(t)N^{-1} + \bigcirc(N^{-2})$$

$$F_{lik}(x) = G_{\mathcal{X}}(x) + -\frac{1}{2}(2n+1)xg_{\mathcal{X}}(x)N^{-1} + \bigcirc(N^{-2}) \tag{4.32}$$

where $g_{\mathcal{X}}(t)$ and $G_{\mathcal{X}}(x)$ are the density and distribution functions of a $\mathcal{X}^2_1$ variable respectively.

Score+search methods

The idea behind this other approach consists in defining a measure to evaluate each candidate Gaussian network (i.e. the *score*) and to use a method to search in the space of possible structures the one with the best score (i.e. the *search*).

All the search methods discussed for Bayesian networks can also be applied for Gaussian networks. In Section 4.3.4 two scores called BIC and K2 were introduced, as well as two search procedures called B Algorithm and local search. Variants of these two search strategies could also be applied for Gaussian networks.

Regarding the score metrics for the continuous domain, different score metrics can be found in the literature in order evaluate how accurately a Gaussian network represents data dependencies. We will discuss the use of two types of them: the penalized maximum likelihood metric and Bayesian scores.

Penalized maximum likelihood: If $L(D \mid S, \boldsymbol{\theta})$ is the likelihood of the database $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ given a Gaussian network model $M = (S, \boldsymbol{\theta})$, then we have that:

$$L(D \mid S, \boldsymbol{\theta}) = \prod_{r=1}^{N} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi v_i}} e^{-\frac{1}{2v_i}(x_{ir} - mi - \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_{jr} - m_j))^2}. \tag{4.33}$$

The maximum likelihood estimates for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$, namely $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_n)$, are obtained either by maximizing $L(D \mid S, \boldsymbol{\theta})$ or also by maximizing the expression $\ln L(D \mid S, \boldsymbol{\theta})$. The definition of the latter is as follows:

$$\ln L(D \mid S, \boldsymbol{\theta}) =$$

$$\sum_{r=1}^{N} \sum_{i=1}^{n} [- \ln(\sqrt{2\pi v_i}) - \frac{1}{2v_i}(x_{ir} - mi - \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_{jr} - m_j))^2] \tag{4.34}$$

where $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_n)$ are the solutions of the following equation system:

$$\begin{cases} \frac{\partial}{\partial m_i} \ln L(D \mid S, \boldsymbol{\theta}) = 0 & i = 1, \ldots, n \\ \frac{\partial}{\partial v_i} \ln L(D \mid S, \boldsymbol{\theta}) = 0 & i = 1, \ldots, n \\ \frac{\partial}{\partial b_{ji}} \ln L(D \mid S, \boldsymbol{\theta}) = 0 & j = 1, \ldots, i-1 \text{ and } X_j \in \boldsymbol{Pa}_i \end{cases} \tag{4.35}$$

As proved in [Larrañaga et al., 2001], the maximum likelihood estimates for $\boldsymbol{\theta}_i = (m_i, b_{ji}, v_i)$ with $i = 1, \ldots, n$ and $j = 1, \ldots, i-1$ and $X_j \in \boldsymbol{Pa}_i$ are obtained as follows:

$$\hat{m}_i = \overline{X}_i$$

$$\hat{b}_{ji} = \frac{S_{X_j X_i}}{S^2_{X_j}}$$

$$\hat{v}_i = S^2_{X_i} - \sum_{X_j \in \boldsymbol{Pa}_i} \frac{S_{X_j X_i}}{S^2_{X_j}} + 2 \sum_{X_j \in \boldsymbol{Pa}_i} \sum_{X_k \in \boldsymbol{Pa}_{i k>j}} \frac{S_{X_j X_k} S_{X_j X_i} S_{X_k X_i}}{S^2_{X_j} S^2_{X_k}} \tag{4.36}$$

where $\overline{X}_i = \frac{1}{N}\sum_{r=1}^{N} x_{ir}$ is the sample mean of variable X_i, $S^2_{X_j} = \frac{1}{N}\sum_{r=1}^{N}\left(x_{jr} - \overline{X}_j\right)^2$ denotes the sample variance of variable X_j, and $S_{X_j X_i} = \frac{1}{N}\sum_{r=1}^{N}\left(x_{jr} - \overline{X}_j\right)\left(x_{ir} - \overline{X}_i\right)$ denotes the sample covariance between variables X_j and X_i. Note that in the case of $\boldsymbol{Pa}_i = \emptyset$, the variance corresponding to the i^{th} variable becomes only dependent on the value $S^2_{X_i}$, as all the rest of the terms in the formula become 0.

As stated before, a general formula for a penalized maximum likelihood score is

$$\sum_{r=1}^{N}\sum_{i=1}^{n}\left[-\ln(\sqrt{2\pi v_i})-\frac{1}{2v_i}\left(x_{ir}-mi-\sum_{x_j\in \boldsymbol{pa}_i} b_{ji}(x_{jr}-m_j)\right)^2\right]-f(N)dim(S). \tag{4.37}$$

The number of parameters, $dim(S)$, required to fully specify a Gaussian network model with a structure given by S can be obtained using the following formula:

$$dim(S)=2n+\sum_{i=1}^{n}\mid \boldsymbol{Pa}_i\mid . \tag{4.38}$$

In fact, for each variable X_i we need to compute its mean, μ_i, its conditional variance, v_i, and its regression coefficients, b_{ji}. The comments on $f(N)$ in Section 3.3.2 are also valid here.

Bayesian scores: The Bayesian Dirichlet equivalent metric (BDe) [Heckerman et al., 1995] has a continuous version for Gaussian networks [Geiger and Heckerman, 1994] called Bayesian Gaussian equivalence (BGe). This metric has the property of being score equivalent. As a result, two Gaussian networks that are isomorphic –i.e. they represent the same conditional independence and dependence assertions– will always obtain the same score.

The metric is based upon the fact that the normal–Wishart distribution is conjugate with regard to the multivariate normal. This fact allows us to obtain a closed formula for the computation of the marginal likelihood of the data once given the structure.

[Geiger and Heckerman, 1994] proved that the marginal likelihood for a general Gaussian network can be computed using the following formula:

$$L(D\mid S)=\prod_{i=1}^{n}\frac{L\left(D^{X_i\cup \boldsymbol{Pa}_i}\mid S_c\right)}{L\left(D^{\boldsymbol{Pa}_i}\mid S_c\right)} \tag{4.39}$$

where each term is on the form given in Equation 4.40, and where $D^{X_i\cup \boldsymbol{Pa}_i}$ is the database D restricted to the variables $X_i\cup \boldsymbol{Pa}_i$.

Combining the results provided by the theorems given in [de Groot, 1970, Geiger and Heckerman, 1994] we obtain:

$$L(D\mid S_c)=(2\pi)^{-\frac{nN}{2}}\left(\frac{\nu}{\nu+N}\right)^{\frac{n}{2}}\frac{c(n,\alpha)}{c(n,\alpha+N)}|T_0|^{\frac{\alpha}{2}}|T_N|^{-\frac{\alpha+N}{2}} \tag{4.40}$$

where $c(n,\alpha)$ is defined as

$$c(n,\alpha)=\left[2^{\frac{\alpha n}{2}}\pi^{\frac{n(n-1)}{4}}\prod_{i=1}^{n}\Gamma\left(\frac{\alpha+1-i}{2}\right)\right]^{-1}. \tag{4.41}$$

This result yields a metric for scoring the marginal likelihood of any Gaussian network.

The reader is referred to [Geiger and Heckerman, 1994] for a discussion on the three components of the user's prior knowledge that are relevant for the learning in Gaussian networks: (1) the prior probabilities $p(S)$, (2) the parameters α and ν, and (3) the parameters $\boldsymbol{\mu}_0$ and T_0.

4.5 Estimation of distribution algorithms for inexact graph matching

After having introduced the notation for EDAs in Sections 4.2.1 and 4.3.1, we will define the way of solving the inexact graph matching problem using any of the EDA approaches introduced so far.

4.5.1 Discrete domains

The representation of individuals that will be used in this section is the one proposed in the second representation of individuals in Section 3.3. The permutation-based representation will be used when applying continuous EDAs, as this representation is more suited for them –a deeper explanation of this is given in Section 4.5.2. When using discrete EDAs, a permutation-based representation will add an extra step for translating the individual to the solution it symbolizes before the individual can be evaluated applying the fitness function, with the consequent lack of performance. In the second representation of Section 3.3 we have individuals that contain directly the solution they symbolize, and therefore the fitness function is applied directly.

Let $G_M = (V_M, E_M)$ be the model graph, and $G_D = (V_D, E_D)$ the data graph obtained from a segmented image. The size of the individuals will be $n = |V_D|$, that is, $\boldsymbol{X} = (X_1, \ldots, X_{|V_D|})$ is a n-dimensional variable where each of its components can take $|V_M|$ possible values (i.e. in our case $r_i = |V_M|$ for $i = 1, \ldots, |V_D|$). We denote by $x_i^1, \ldots, x_i^{|V_M|}$ the possible values that the i^{th} variable, X_i, can have.

In the same way, for the unrestricted discrete distributions θ_{ijk}, the range of i, j and k for graph matching using the representation of individuals proposed is as follows: $i = 1, \ldots, |V_D|$, $k = 1, \ldots, |V_M|$, and $j = 1, \ldots, q_i$, where $q_i = |V_M|^{npa_i}$ and npa_i denotes the number of parents of X_i.

4.5.1.1 Estimating the probability distribution

We propose three different EDAs to be used to solve inexact graph matching problems in the discrete domain. The fact that the difference in behavior between algorithms is to a large extent due to the complexity of the probabilistic structure that they have to build, these three algorithms have been selected so that they are representatives of the many different types of discrete EDAs. Therefore, these algorithms can be seen as representatives of the three categories of EDAs introduced in Section 4.3: (1) UMDA [Mühlenbein, 1998] is an example of an EDA that considers no interdependencies between the variables (i.e. the learning is only parametrical, not structural). This assumption can be allowed in the case of inexact graph matching depending on the complexity of the problem, and mostly depending on $|V_M|$ and $|V_D|$. It is important to realize that a balance between cost and performance must be achieved, and therefore, in some complex problems the use of UMDA can be justified when trying to shorten the computation cost. Nevertheless, other algorithms that do not require such assumptions should return a better result if fast computation is not essential. (2) MIMIC [de Bonet et al., 1997] is an example that belongs to the category of pairwise dependencies. Therefore, there is an extra task in MIMIC, which is the construction of the probabilistic graphical model that represents the pairwise dependencies. As a result, when applying this algorithm to graph matching, the computation time for the structural learning is proportional to the number of vertices of the graphs ($|V_M|$ and $|V_D|$). This shows again that due to the higher cost that MIMIC has UMDA can be used in order to obtain best results in a fixed period of time. (3) EBNA [Etxeberria and Larrañaga, 1999] is an example of the category of EDAs where multiple interdependencies are allowed between the variables, on which the structural learning is even more complex than in the two previous algorithms. The models constructed with EDAs describe the interdependencies between the different variables more accurately than those of MIMIC (and, obviously, better than with UMDA). Nevertheless, as the size of the graphs to match has a direct influence on the complexity of the Bayesian network to be build (the complexity and the computation time increase exponentially with $|V_M|$ and $|V_D|$), the use of other simpler probabilistic graphical models –which restrict the number of parents that a variable can have in the probabilistic structure– such as the two proposed above is justified.

Finally, a last remark about the fact that within the same algorithm there are sometimes different techniques to find the probabilistic structure that best suits the n–dimensional probabilistic model. In the case of Bayesian networks for instance, there are two ways of building the graph: by detecting dependencies, and through score and search [Buntine, 1996, de Campos, 1998, Heckerman, 1995, Krause, 1998, Sangüesa and Cortés, 1998]. These different possibilities can also have an influence on the structure estimated at each generation to represent the model. In the case of EBNA a score+search method using the *BIC* score is used, although any other score could also be used.

4.5.1.2 Adapting the simulation scheme to obtain correct individuals

Section 3.3.3 introduces the conditions that have to be satisfied to consider an individual as *correct* for the particular graph matching problems that we are considering in this thesis. In the same section we also showed that, in order to consider a solution as correct, the only condition to check is that all the vertices in graph G_M contain at least a matching, that is, that every vertex of G_M appears in the individual at least once. If we include a method in EDAs to ensure that this condition will be satisfied by each individual sampled, this would prevent the algorithm from generating incorrect individuals like the one in Figure 3.3b.

There are at least three ways of facing the problem of the existence of incorrect individuals in EDAs: controlling directly the simulation step, correction a posteriori, and changing the fitness function. The first technique can only be applied to EDAs, as it is based on modifying the previously estimated probability distribution and therefore it cannot be put into practice on other types of algorithms. The last two techniques can be applied to EDAs and other heuristics that deal with constraints. In [Michalewicz, 1992, Michalewicz and Schoenauer, 1996] we can find examples of GAs applied to problems where individuals must satisfy specific constraints for the problem.

Next, we discuss some examples of these techniques to control the generation of the individuals. Even if all of them are presented as a solution to graph matching, they can equally be applied to any other problem where individuals must satisfy any constraints.

Controlling directly the simulation step

Up to now in most of the problems where EDAs have been applied no constraints had to be taken into account. This is the reason why very few articles about modifying the simulation step for this purpose can be found –some of them are for instance [Bengoetxea et al., 2000, 2002a] and [Santana and Ochoa, 1999]. In our inexact graph matching problem the nature of the individuals is special enough to require a modification of this simulation step. Two different ways of modifying the simulation step are introduced in this section.

However, it is important to note that altering the probabilities at the simulation step, whichever the way, implies that the learning of the algorithm is also denaturalized somehow. It is therefore very important to make sure that the manipulation is only performed to *guide* the generation of potentially incorrect individuals towards correct ones.

- **Last Time Manipulation (LTM): forcing the selection of values still to appear only at the end.** This method consists in not altering the simulation step during the generation of the individual until the number of vertices of G_M remaining to match and the number of variables to be simulated in the individual are equal. For instance, this could happen when three vertices of G_M have not been matched yet and the values of the last three variables have to be calculated for an individual. In this case, we will force the simulation step so that only these three values could be sampled in the next variable.

  In order to force the next variable of the individual to take only one of the values that have not still appeared, the value of the probabilities that are used to perform the simulation will be changed. In this way, we will set the probability of all the values already appeared in the individual to 0, and the probabilities of the values not still appeared will be modified accordingly. More formally, the procedure to generate an individual will follow the order $\boldsymbol{\pi} = (\pi(1), \pi(2), \ldots, \pi(|V_D|))$, that is, all the variables will be instantiated in the following order: $\left(X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(|V_D|)}\right)$. If we are instantiating the m^{th} variable (i.e. we are sampling the variable $X_{\pi(m)}$), the following definitions apply: let be $VNO(V_M)^m = \{u_M^i \in V_M \mid \nexists X_j \in \boldsymbol{X}\ j \in \{\pi(1), \ldots, \pi(m-1)\},\ X_j = i\}$ the set that contains all the vertices of G_M not yet matched in the individual in the previous $m-1$ steps (VNO stands for Vertices Not Obtained), $vns^m = |V_D| - m$ is the number of variables still to be simulated, $\theta_{\pi(m)lk}$ is the probability of the variable $X_{\pi(m)}$ to take the value $x_{\pi(m)}^k$ (its k^{th} value) knowing that its parents are already on their l^{th} possible combination of values (as $\boldsymbol{\pi}$ follows an ancestral ordering, we know that the parent variables of $X_{\pi(m)}$ have already been instantiated in the previous $m-1$ steps), and $P_{Indiv}^m = \sum_{k \mid u_M^k \in V_M \setminus VNO(V_M)^m} \theta_{\pi(m)lk}$. With these definitions, following this method we will only modify the $\theta_{\pi(m)lk}$ values when the condition $|VNO(V_M)^m| = vns^m$ is satisfied. When this is the case, the probability for the value $x_{\pi(m)}^k$ to appear next in the variable $X_{\pi(m)}$ of the individual knowing that its parents are in the l^{th} combination of values, $\theta_{\pi(m)lk}^*$, will be adapted as follows:

$$\theta_{\pi(m)lk}^* = \begin{cases} \theta_{\pi(m)lk} \cdot \frac{1}{1-P_{Indiv}^m} & \text{if } u_M^{\pi(m)} \in VNO(V_M)^m \\ 0 & \text{otherwise.} \end{cases} \tag{4.42}$$

  Once the probabilities have been modified, it is guaranteed that the only values assigned to $X_{\pi(m)}$ will be one of the vertices of V_M not still obtained (a vertex from $VNO(V_M)^m$), as the probability to obtain next any vertex from $V_M \setminus VNO(V_M)^m$ has been set to 0. This modification of the $\theta_{\pi(m)lk}$ has to be repeated for the rest of the variables of the individual, but taking into account that in the next step there is one more value which probability has to be set to 0, and thus P_{Indiv}^m must be computed again. Following this

```
Adaptation 1 of PLS: Last Time Manipulation (LTM)
Definitions
    $vns^m$: number of variables not simulated before the $m^{th}$ step
    $VNO(V_M)^m$: set of vertices of $V_M$ not matched by simulation before
        the $m^{th}$ step
    $P^m_{Indiv}$: sum of the probabilities to appear next in the individual any vertex
        of $V_M$ already matched before the $m^{th}$ step
    $\theta_{ijk}$: probability to appear the value $k$ next in the variable $X_i$ knowing that
        the values defined in the individual for its parents are on their $j^{th}$
        possible combination of values
Procedure
    Find an ancestral ordering, $\boldsymbol{\pi}$, of the nodes in the Bayesian network
    For $m = \pi(1), \pi(2), \ldots, \pi(|V_D|)$ (number of variables, to sample following
    the ancestral ordering $\boldsymbol{\pi}$)
        If $(|VNO(V_M)^m| == vns^m)$
            For $k = 1, 2, \ldots, |V_M|$ (number of values for each variable)
                Modify the probabilities: the modified probability for the
                value $k$ to appear next in the variable $X_{\pi(m)}$ of the
                individual for the $l^{th}$ combination of values of its
                parents, $\theta^*_{\pi(m)lk}$, is
                $\theta^*_{\pi(m)lk} = \begin{cases} \theta_{\pi(m)lk} \cdot \frac{1}{1-P^m_{Indiv}} & \text{if } u_M^{\pi(m)} \in VNO(V_M)^m \\ 0 & \text{otherwise} \end{cases}$
                where $P^m_{Indiv} = \sum_{k \mid u^k_M \in V_M \setminus VNO(V_M)^m} \theta_{ilk}$, and $l$ is the
                combination of values of the parent variables of $X_{\pi(m)}$
                (which have been previously instantiated in the previous
                $m-1$ iterations)
            $X_{\pi(m)} \leftarrow$ generate a value from $\theta^*_{\pi(m)lk} = p\left(X_{\pi(m)} = k \mid \boldsymbol{pa}^l_{\pi(m)}\right)$
        Else
            $X_{\pi(m)} \leftarrow$ generate a value from $\theta_{\pi(m)lk} = p\left(X_{\pi(m)} = k \mid \boldsymbol{pa}^l_{\pi(m)}\right)$
```

Figure 4.16: Pseudocode for Last Time Manipulation (LTM).

method, at the last step ($m = |V_D|$), only one value v will have its probability set to $\theta^*_{\pi(m)lv} = 1$ and for the rest of the values $\theta^*_{\pi(m)lw} = 0 \quad \forall w \neq v$. Therefore, the only value that will be assigned to the variable $X_{|V_D|}$ will be v.

This technique does not modify the probabilities of the variables in any way until $|VNO(V_M)^m| = vns^m$. Therefore, the simulation step will remain as it is, without any external manipulation unless the latter condition is satisfied. However, when that condition is satisfied the method will modify the actual distribution that the values will follow.

Figure 4.16 shows the pseudocode of this first adaptation of PLS. A detailed example of LTM can also be found in Appendix B.

- **All Time Manipulation (ATM): increasing the probability of the values not appeared from the beginning.** This second technique is another way of manipulating the probabilities of the values for each variable within the individual, but this time the manipulation takes place not only at the end but from the beginning of the generation of the individual. The value of the probabilities remains unaltered only after all the possible values of the variables have already appeared in the individual (that is, when $VNO(V_M) = \emptyset$).

For this, again the order of sampling the variables $\boldsymbol{\pi}$ will be followed, instantiating them in the same order $\left(X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(|V_D|)}\right)$. Every step the probabilities of a variable will be modified before its instantiation. The required definitions for the m^{th} step (the sampling of the variable $X_{\pi(m)}$) are as follows: let $|V_D|$ be number of variables of each individual, let also be $VNO(V_M)^m$, vns^m, and $\theta_{\pi(m)lk}$ as defined before. The latter probability will be modified with this method obtaining the new $\theta^*_{\pi(m)lk}$ as follows:

$$\theta^*_{\pi(m)lk} = \begin{cases} \theta_{\pi(m)lk} \cdot \frac{K - P^m_{Indiv}}{K \cdot \left(1 - P^m_{Indiv}\right)} & \text{if } u^i_M \in VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| \neq vns^m \\ \frac{\theta_{\pi(m)lk}}{K} & \text{if } u^i_M \notin VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| \neq vns^m \\ \theta_{\pi(m)lk} \cdot \frac{1}{1 - P^m_{Indiv}} & \text{if } u^i_M \in VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| = vns^m \\ 0 & \text{if } u^i_M \notin VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| = vns^m \end{cases} \tag{4.43}$$

where $K = \left\lceil \frac{N - vns^m}{vns^m - |VNO(V_M)^m|} \right\rceil$, and $P^m_{Indiv} = \sum_{u^k_m \in V_M \setminus VNO(V_M)^m} \theta_{\pi(m)lk}$.

In fact, the two last cases are defined only to avoid the division by zero problem, but these two cases can also be calculated when using limits, as the $|VNO(V_M)^m| = vns^m$ case can be understood as the limit when $K \longrightarrow \infty$:

$$\begin{aligned} |VNS(V_D)| &= |VNO(V_M)| \Rightarrow \\ \theta^*_{ijk} &= \lim_{K \to \infty} \left(\theta_{ijk} \cdot \frac{K - P_{Indiv}}{K \cdot (1 - P_{Indiv})} \right) = \\ &= \theta_{ijk} \cdot \lim_{K \to \infty} \left(\frac{1 - \frac{P_{Indiv}}{K}}{1 - P_{Indiv}} \right) = \theta_{ijk} \cdot \frac{1}{1 - P_{Indiv}} \\ \text{and} & \\ |VNS(V_D)| &= |VNO(V_M)| \Rightarrow \\ \theta^*_{ijk} &= \lim_{K \to \infty} \left(\frac{\theta_{ijk}}{K} \right) = 0 \end{aligned}$$

The reason to modify the probabilities in such a manner is that at the beginning, when vns^m is much bigger than $|VNO(V_M)^m|$, there is still time for all the values to appear in the individual, and thus the probabilities are not modified very much. Only when $|VNO(V_M)^m|$ starts to be very close to vns^m will the effect of the manipulation be stronger, meaning that there are not much variables to be instantiated regarding the values not appeared yet on the individual. Finally, when $|VNO(V_M)^m| = vns^m$, there is no chance to leave the probabilities as they are, and only the values not appeared yet have to be selected. For this, the probabilities of the values already appeared are set to 0, and the other ones are modified in the same way as in the previous method.

```
Adaptation 2 of PLS: All Time Manipulation (ATM)
Definitions
    $vns^m$: number of variables not simulated before the $m^{th}$ step.
    $VNO(V_M)^m$: set of vertices of $V_M$ not matched by simulation before
        the $m^{th}$ step
    $P^m_{Indiv}$: sum of the probabilities to appear next in the individual any
        vertex of $V_M$ already matched before the $m^{th}$ step
    $\theta_{ijk}$: probability to appear the value $k$ next in the variable $X_i$
        knowing that the values defined in the individual for its parents
        are on their $j^{th}$ possible combination of values.
Procedure
    Find an ancestral ordering, $\boldsymbol{\pi}$, of the nodes in the Bayesian network
    For $m = \pi(1), \pi(2), \ldots, \pi(|V_D|)$ (number of variables, to sample
    following the ancestral ordering $\boldsymbol{\pi}$)
        If $(|VNO(V_M)^m| > 0)$
            For $k = 1, 2, \ldots, |V_M|$ (number of values for each variable)
                Modify the probabilities: the modified probability for
                the value $k$ to appear next in the variable $X_{\pi(m)}$ of
                the individual for the $l^{th}$ combination of values of its
                parents, $\theta^*_{\pi(m)lk}$, is
                $\theta^*_{\pi(m)lk} = \begin{cases} \theta_{\pi(m)lk} \cdot \frac{K - P^m_{Indiv}}{K \cdot \left(1 - P^m_{Indiv}\right)} & \text{if } u^i_M \in VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| \neq vns^m \\ \frac{\theta_{\pi(m)lk}}{K} & \text{if } u^i_M \notin VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| \neq vns^m \\ \theta_{\pi(m)lk} \cdot \frac{1}{1 - P^m_{Indiv}} & \text{if } u^i_M \in VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| = vns^m \\ 0 & \text{if } u^i_M \notin VNO(V_M)^m \text{ and} \\ & |VNO(V_M)^m| = vns^m \end{cases}$
                where $K = \left\lceil \frac{N-m}{m-n} \right\rceil$, $l$ is the combination of values of the
                parent variables of $X_{\pi(m)}$ (which have been previously
                instantiated in the previous $m-1$ iterations),
                and $P^m_{Indiv} = \sum_{k \mid u^k_M \in V_M \setminus VNO(V_M)^m} \theta_{ilk}$
            $X_{\pi(m)} \leftarrow$ generate a value from $\theta^*_{\pi(m)lk} = p\left(X_{\pi(m)} = k \mid \boldsymbol{pa}^l_{\pi(m)}\right)$
        Else
            $X_{\pi(m)} \leftarrow$ generate a value from $\theta_{\pi(m)lk} = p\left(X_{\pi(m)} = k \mid \boldsymbol{pa}^l_{\pi(m)}\right)$
```

Figure 4.17: Pseudocode for All Time Manipulation (ATM).

Figure 4.17 shows the pseudocode of this second adaptation of the simulation. A detailed example of ATM can also be found in Appendix B.

This second technique modifies the probabilities nearly from the beginning, giving more chance to the values not already appeared, but it also takes into account the probabilities learned by the Bayesian network in the learning step. It does not modify the probabilities in any way when $|VNO(V_M)^m| = 0$, that is, when all the values have already appeared in the individual.

Correction a posteriori

This technique is completely different from the ones proposed before, as it is not based on modifying the probabilities generated by the algorithm at all: the idea is to correct the individuals that do not contain an acceptable solution to the problem after they have been completely generated. In order to do this correction, once the individual has been completely generated and has been identified as not correct ($|VNO(V_M)^{|V_D|}| > 0$), a variable which contains a value that appears more than once in the individual is chosen randomly and substituted by one of the missing values. This task is performed $|VNO(V_M)^{|V_D|}|$ times, that is, until the individual is correct.

The fact that no modification is done at all in the learned probabilities means that this method does not demerit the learning process, and thus the learning process is respected as when using PLS. As the generation of the individuals is not modified at all with respect to PLS, the only manipulation occurs on the wrong individuals, and the algorithm can be supposed to require less generations to converge to the final solution. Furthermore, this method can also be used with other evolutionary computation techniques such as GAs.

Changing the fitness function

This last method is not based neither on modification of the probabilities during the process of the generation of the new individuals nor in adapting them later before adding them to the population. Instead, the idea is completely different and consists in applying a penalization in the fitness value of each individual.

The penalization has to be designed specifically for each problem in order to avoid unexpected results. For instance, on the experiments carried out with this technique and commented later in Section 6.2, the penalization has been defined as follows: if $f(\boldsymbol{x})$ is the value obtained by the fitness function for the individual $\boldsymbol{x}= (x_1, \ldots x_{|V_D|})$, and if $|VNO(V_M)^{|V_D|}|$ is the number of vertices of G_M not present in the individual, the modified fitness value, $f^*(\boldsymbol{x})$ will be changed as follows:

$$f^*(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{|VNO(V_M)^{|V_D|}| + 1}. \tag{4.44}$$

Another important difference regarding the other methods to control the generation of individuals explained so far is that the penalization does allow the generation of incorrect individuals, and therefore these will still appear in the successive generations. This aspect needs to be analyzed for every problem when a penalization is applied. Nevertheless, as these incorrect individuals will be given a lower fitness value, it is expected that their number will be reduced in future generations. It is therefore important to ensure that the penalization applied to the problem is strong enough. On the other hand, the existence of these individuals can be regarded as a way to avoid local maxima, expecting that, starting from them, fittest correct individuals would be found.

4.5.2 Continuous domains

Similarly as in Section 4.5.1, we will define the graph matching problem using a particular notation for continuous EDAs.

The selected representation of individuals for continuous domains was introduced previously in Section 3.3.2. These individuals are formed of only continuous values, and as already explained no combination of values from both the discrete and continuous domains is considered in this thesis. As explained in that section, in order to compute the fitness value of each individual, a previous translation step is required from the continuous representation to a permutation-based discrete one (as shown in Figure 3.2), and finally in a second step from there to a representation in the discrete domain like the one explained in Section 3.3 –the latter procedure has already been described in Figure 3.1. All these translations have to be performed for each individual, and therefore these operations result in an overhead in the execution time which makes the evaluation time longer for the discrete domain when using this method. It is important to note that the meaning of the values for each variable is not directly linked to the solution the individual symbolizes, and therefore we will not refer directly to graphs G_M and G_D in this section.

Section 3.3.2 defined that the size of these continuous individuals is $n = |V_D|$, that is, there are $\boldsymbol{X} = \left(X_1, \ldots, X_{|V_D|}\right)$ continuous variables, each of them having any value in a range of (-100, 100) for instance. We denote by x_i the values that has the i^{th} variable, X_i.

Estimating the density function

We propose different continuous EDAs belonging to different categories to be used in inexact graph matching. As explained in the previous sections, these algorithms are expected to increase their complexity when more complex learning algorithms are used and when more complicated structures are allowed to be learned. Again, these algorithms should be regarded as representatives of their categories introduced in Section 4.4.1: (1) UMDA$_c$ [Larrañaga et al., 2000, 2001] as an example of an EDA that considers no interdependencies between the variables; (2) MIMIC$_c$ [Larrañaga et al., 2000, 2001] which is an example that belongs to the category of pairwise dependencies; and finally, (3) EGNA$_{BGe}$ and EGNA$_{BIC}$ [Larrañaga et al., 2000, Larrañaga and Lozano, 2001] as an example of the category of EDAs where multiple interdependencies are allowed between the variables, where no limits are imposed on the complexity of the model to learn. It is important to note that any other algorithm mentioned in Section 4.4.1 could perfectly have been chosen as a representative of its category instead of the ones we have selected.

It is important to note that even if EGNA$_{BGe}$ is expected to obtain better results, the fact of not setting limits to the number of interdependencies that a variable can have could also mean that the best structure is a simple one. For instance, a structure learned with EGNA$_{BGe}$ could perfectly contain at most pairwise dependencies. This fact is completely dependent on the complexity of the graph matching problem applied in our case. This means that in such cases the difference in results obtained with EGNA$_{BGe}$ and MIMIC$_c$ would not show significant differences, while the computation time in EGNA$_{BGe}$ will be significantly higher.

Adapting the simulation scheme to obtain correct individuals

In the discrete domain the need to introduce adaptations in the simulation arises because of the additional constraint in our proposed graph matching problems for all the solutions to contain all the possible values of V_M.

However, in the case of the continuous domain no such restriction exists, and the fact of using a procedure to convert each continuous individual to a permutation of discrete values ensures that the additional constraint is always satisfied for any continuous individual. Therefore, all the possible continuous individuals in the search space do not require to any adaptation in the simulation step, and therefore no measures need to be introduced at this point.